Refine Search

Search Results -

Terms	Documents
L3 and @pd > 20040603	0

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L4

Refine Search | Recall Text | Clear | Interrupt

Search History

DATE: Monday, May 29, 2006 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=NO; OP=OR			
L4	L3 and @pd > 20040603	0	L4
L3	L2 and @PY<2003	71	L3
L2	L1 AND (717/$$$.ccls. OR 705/$$$.ccls. OR 709/$$$.ccls.)	162	L2
L1	Workflow ADJ system	270	L1

END OF SEARCH HISTORY

Refine Search

Search Results -

Terms	Documents
L3 and @pd > 20050801	0

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L5

Refine Search

Recall Text | Clear | Interrupt

Search History

DATE: Monday, May 29, 2006 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=NO; OP=OR			
L5	L3 and @pd > 20050801	0	L5
L4	L3 and @pd > 20040603	0	L4
L3	L2 and @PY<2003	71	L3
L2	L1 AND (717/$$$.ccls. OR 705/$$$.ccls. OR 709/$$$.ccls.)	162	L2
L1	Workflow ADJ system	270	L1

END OF SEARCH HISTORY

Refine Search

Search Results -

Terms	Documents
(dynamic ADJ compilation) and workflow	1

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L7

Refine Search | Recall Text | Clear | Interrupt

Search History

DATE: Monday, May 29, 2006 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=NO; OP=OR			
L7	(dynamic ADJ compilation) and workflow	1	L7
L6	dymanic ADJ compilation and workflow	0	L6
L5	L3 and @pd > 20050801	0	L5
L4	L3 and @pd > 20040603	0	L4
L3	L2 and @PY<2003	71	L3
L2	L1 AND (717/$$$.ccls. OR 705/$$$.ccls. OR 709/$$$.ccls.)	162	L2
L1	Workflow ADJ system	270	L1

END OF SEARCH HISTORY

Terms used **JAVA backward compatibility** Found **3,527** of **177,263**

Sort results by: relevance
Display results: expanded form
Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next
Best 200 shown Relevance scale

1 Making the future safe for the past: adding genericity to the Java programming language
Gilad Bracha, Martin Odersky, David Stoutamire, Philip Wadler
October 1998 **ACM SIGPLAN Notices , Proceedings of the 13th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '98**, Volume 33 Issue 10
Publisher: ACM Press
Full text available: pdf(1.91 MB) Additional Information: full citation, abstract, references, citings, index terms

We present GJ, a design that extends the Java programming language with generic types and methods. These are both explained and implemented by translation into the unextended language. The translation closely mimics the way generics are emulated by programmers: it erases all type parameters, maps type variables to their bounds, and inserts casts where needed. Some subtleties of the translation are caused by the handling of overriding.GJ increases expressiveness and safety: code utilizing generic ...

2 Compatible genericity with run-time types for the Java programming language
Robert Cartwright, Guy L. Steele
October 1998 **ACM SIGPLAN Notices , Proceedings of the 13th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '98**, Volume 33 Issue 10
Publisher: ACM Press
Full text available: pdf(1.97 MB) Additional Information: full citation, abstract, references, citings, index terms

The most serious impediment to writing substantial programs in the Java™ programming language is the lack of a *gentricity* mechanism for abstracting classes and methods with respect to type. During the past two years, several research groups have developed Java extensions that support various forms of genericity, but none has succeeded in accommodating general type parameterization (akin to Java arrays) while retaining compatibility with the existing. Java Virtual Machine. In thi ...

3 MultiJava: modular open classes and symmetric multiple dispatch for Java
Curtis Clifton, Gary T. Leavens, Craig Chambers, Todd Millstein
October 2000 **ACM SIGPLAN N tices , Pr ceedings of the 15th ACM SIGPLAN conference n Object-oriented pr gramming, systems, languages, and applications OOPSLA '00**, Volume 35 Issue 10
Publisher: ACM Press

Full text available: pdf(196.92 KB) Additional Information: full citation, abstract, references, citings, index terms

We present MultiJava, a backward-compatible extension to Java supporting *open classes* and *symmetric multiple dispatch*. Open classes allow one to add to the set of methods that an existing class supports without creating distinct subclasses or editing existing code. Unlike the "Visitor" design pattern, open classes do not require advance planning, and open classes preserve the ability to add new subclasses modularly and safely. Multiple dispatch offers several well-known advantages ...

4 Technical correspondence: Generics in Java and C++: a comparative model

Debasish Ghosh
May 2004 **ACM SIGPLAN Notices**, Volume 39 Issue 5
Publisher: ACM Press
Full text available: pdf(868.17 KB) Additional Information: full citation, references

5 Javari: adding reference immutability to Java

Matthew S. Tschantz, Michael D. Ernst
October 2005 **ACM SIGPLAN Notices , Proceedings of the 20th annual ACM SIGPLAN conference on Object oriented programming, systems, languages, and applications OOPSLA '05**, Volume 40 Issue 10
Publisher: ACM Press
Full text available: pdf(345.67 KB) Additional Information: full citation, abstract, references, index terms

This paper describes a type system that is capable of expressing and enforcing immutability constraints. The specific constraint expressed is that the abstract state of the object to which an immutable reference refers cannot be modified using that reference. The abstract state is (part of) the transitively reachable state: that is, the state of the object and all state reachable from it by following references. The type system permits explicitly excluding fields from the abstract state of an ob ...

Keywords: Java, Javari, assignable, immutability, mutable, readonly, type system, verification

6 Converting java programs to use generic libraries

Alan Donovan, Adam Kiežun, Matthew S. Tschantz, Michael D. Ernst
October 2004 **ACM SIGPLAN Notices , Proceedings of the 19th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '04**, Volume 39 Issue 10
Publisher: ACM Press
Full text available: pdf(1.18 MB) Additional Information: full citation, abstract, references, index terms

Java 1.5 will include a type system (called JSR-14) that supports parametric polymorphism, or generic classes. This will bring many benefits to Java programmers, not least because current Java practice makes heavy use of logically-generic classes, including container classes.

Translation of Java source code into semantically equivalent JSR-14 source code requires two steps: parameterization (adding type parameters to class definitions) and instantiation (a ...

Keyw rds: JSR-14, Java 1.5, Java 5, generic types, instantiation types, parameterized types, parametric polymorphism, raw types, type inference

7 Regular contributions: Adapting Tomasulo's algorithm for bytecode folding based Java processors

M. Watheq El-Kharashi, Fayez Elguibaly, Kin F. Li

December 2001 **ACM SIGARCH C mputer Archi tecture News**, Volume 29 Issue 5

Publisher: ACM Press

Full text available: pdf(801.36 KB) Additional Information: full citation, abstract, references, index terms

A novel processor architecture for hardware execution of Java bytecodes is presented. Stack dependency is resolved by the use of a hardware bytecode folding algorithm coupled with Tomasulo's scheduling algorithm. In this paper, we present a framework for adapting Tomasulo's algorithm for bytecode folding based Java processors. We discuss a set of architectural features that are tailored for Java execution as well as for general-purpose Java-independent codes. A comprehensive example is included ...

Keywords: dynamic scheduling, instruction shelving, java, java bytecode folding, java processors, java stack folding, java virtual machine, register renaming, reservation stations, stack processors, tomasulo's algorithm

8 Real-time convergence of Ada and Java™

Ben Brosgol, Brian Dobbing

September 2001 **ACM SIGAda Ada Letters , Proceedings of the 2001 annual ACM SIGAda international conference on Ada SIGAda '01**, Volume XXI Issue 4

Publisher: ACM Press

Full text available: pdf(191.98 KB) Additional Information: full citation, abstract, references, citings, index terms

Two independent recent efforts have defined extensions to the Java platform that intend to satisfy real-time requirements. This paper summarizes the major features of these efforts, compares them to each other and to Ada 95's Real-Time Annex, and argues that their convergence with Ada95 may serve to complement rather than compete with Ada in the real-time domain.

Keywords: Ada, Java, Real-Time, asynchrony, garbage collection, scheduling, threads

9 Parametric polymorphism in Java: an approach to translation based on reflective features

Mirko Viroli, Antonio Natali

October 2000 **ACM SIGPLAN Notices , Proceedings of the 15th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '00**, Volume 35 Issue 10

Publisher: ACM Press

Full text available: pdf(277.16 KB) Additional Information: full citation, abstract, references, citings, index terms

The introduction of parametric polymorphism in Java with translation approaches has been shown to be of considerable interest, allowing the definition of extensions of Java on top of the existing Virtual Machines. Homogeneous translations furthermore, seem to be more useful than heterogeneous, avoiding the continuous increase of library code with redundant information. At this time however, homogeneous approaches aren't as flexible as heterogeneous, with extensions failing to integrate well with ...

10 Automatically generating refactorings to support API evolution

Jeff H. Perkins

September 2005 **ACM SIGSOFT S ftware Engineering Notes , The 6th ACM SIGPLAN-**

SIGSOFT w rkshop n Pr gram an alysis f r software t ols and engineering PASTE '05, Volume 31 Issue 1
Publish r: ACM Press
Full text available: pdf(68.44 KB) Additional Information: full citation, abstract, references, index terms

When library APIs change, client code should change in response, in order to avoid erroneous behavior, compilation failures, or warnings. Previous research has introduced techniques for generating such client refactorings. This paper improves on the previous work by proposing a novel, lightweight technique that takes advantage of information that programmers can insert in the code rather than forcing them to use a different tool to re-express it. The key idea is to replace calls to deprecated me ...

11 Adding type parameterization to the Java language
Ole Agesen, Stephen N. Freund, John C. Mitchell
October 1997 **ACM SIGPLAN Notices , Proceedings of the 12th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '97**, Volume 32 Issue 10
Publisher: ACM Press
Full text available: pdf(2.16 MB) Additional Information: full citation, abstract, references, citings, index terms

Although the Java programming language has achieved widespread acceptance, one feature that seems sorely missed is the ability to use type parameters (as in Ada generics, C++ templates, and ML polymorphic functions or data types) to allow a general concept to be instantiated to one or more specific types. In this paper, we propose parameterized classes and interfaces in which the type parameter may be constrained to either implement a given interface or extend a given class. This design allows t ...

12 Practitioners report: Programming with non-heap memory in the real time specification for Java
Greg Bollella, Tim Canham, Vanessa Carson, Virgil Champlin, Daniel Dvorak, Brian Giovannoni, Mark Indictor, Kenny Meyer, Alex Murray, Kirk Reinholtz
October 2003 **Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**
Publisher: ACM Press
Full text available: pdf(227.11 KB) Additional Information: full citation, abstract, references, index terms

The Real-Time Specification for Java (RTSJ) provides facilities for deterministic, real-time execution in a language that is otherwise subject to variable latencies in memory allocation and garbage collection. A major consequence of these facilities is that the normal Java practice of passing around references to objects in heap memory cannot be used in hard real-time activities. Instead, designers must think carefully about what type of non-heap memory to use and how to transfer data between co ...

Keywords: architecture, programming model, scoped memory

13 Portable resource control in Java
Walter Binder, Jane G. Hulaas, Alex Villazón
October 2001 **ACM SIGPLAN Notices , Proceedings of the 16th ACM SIGPLAN conference on Object oriented programming, systems, languages, and applicati ns OOPSLA '01**, Volume 36 Issue 11
Publisher: ACM Press
Full text available: pdf(307.08 KB) Additional Information: full citation, abstract, references, citings, index terms

Preventing abusive resource consumption is indispensable for all kinds of systems that execute untrusted mobile coee, such as mobile object sytems, extensible web servers,

and web browsers. To implement the required defense mechanisms, some support for resource control must be available: accounting and limiting the usage of physical resources like CPU and memory, and of logical resources like threads. Java is the predominant implementation language for the kind of systems envisaged here, even th ...

Keywords: Java, bytecode rewriting, micro-kernels, mobile object systems, resource control, security

14 Advanced control flow in Java card programming

Peng Li, Steve Zdancewic

June 2004 **ACM SIGPLAN Notices , Proceedings of the 2004 ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools for embedded systems LCTES '04**, Volume 39 Issue 7

Publisher: ACM Press

Full text available: pdf(205.46 KB) Additional Information: full citation, abstract, references, index terms

Java Card technology simplifies the development of smart card applications by providing a high-level programming language similar to Java. However, the master-slave programming model used in current Java Card platform creates control flow difficulties when writing complex card programs, making it inconvenient, tedious, and error-prone to implement Java Card applications. This paper examines these drawbacks of the master-slave model and proposes a concurrent thread model for developing future Jav ...

Keywords: CPS, Java card, continuation, control flow, smart card, trampolined style

15 Parametric polymorphism for Java: a reflective solution

Jose H. Solorzano, Suad Alagić

October 1998 **ACM SIGPLAN Notices , Proceedings of the 13th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '98**, Volume 33 Issue 10

Publisher: ACM Press

Full text available: pdf(1.38 MB) Additional Information: full citation, abstract, references, citings, index terms

A number of inadequacies of existing implementation techniques for extending Java™ with parametric polymorphism are revealed. Homogeneous translations are the most space-efficient but they are not compatible with reflection, some models of persistence, and multiple dispatch. Heterogeneous translations, on the other hand, can potentially produce large amounts of redundant information. Implementation techniques that address these concerns are developed. In languages that support run-time ...

Keywords: language design and implementation, persistence, reflection

16 On type systems for object-oriented database programming languages

Yuri Leontiev, M. Tamer Özsu, Duane Szafron

December 2002 **ACM Computing Surveys (CSUR)**, Volume 34 Issue 4

Publisher: ACM Press

Full text available: pdf(346.87 KB) Additional Information: full citation, abstract, references, index terms

The concept of an object-oriented database programming language (OODBPL) is appealing because it has the potential of combining the advantages of object orientation and database programming to yield a powerful and universal programming language design. A uniform and consistent combination of object orientation and database

programming, however, is not straightforward. Since one of the main components of an object-oriented programming language is its type system, one of the first problems that ar ....

Keywords: OODB, OODBPL, object-oriented database programming language, type checking, typing

17 Featherweight Java: a minimal core calculus for Java and GJ

Atsushi Igarashi, Benjamin C. Pierce, Philip Wadler

May 2001 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 23 Issue 3

Publisher: ACM Press

Full text available: pdf(644.38 KB) Additional Information: full citation, abstract, references, citings, index terms

Several recent studies have introduced lightweight versions of Java: reduced languages in which complex features like threads and reflection are dropped to enable rigorous arguments about key properties such as type safety. We carry this process a step further, omitting almost all features of the full language (including interfaces and even assignment) to obtain a small calculus, Featherweight Java, for which rigorous proofs are not only possible but easy. Featherweight Java bears a similar rela ...

Keywords: Compilation, Java, generic classes, language design, language semantics

18 Parametric polymorphism for software component architectures

Cosmin E. Oancea, Stephen M. Watt

October 2005 **ACM SIGPLAN Notices , Proceedings of the 20th annual ACM SIGPLAN conference on Object oriented programming, systems, languages, and applications OOPSLA '05**, Volume 40 Issue 10

Publisher: ACM Press

Full text available: pdf(375.78 KB) Additional Information: full citation, abstract, references, index terms

Parametric polymorphism has become a common feature of mainstream programming languages, but software component architectures have lagged behind and do not support it. We examine the problem of providing parametric polymorphism with components combined from different programming languages. We have investigated how to resolve different binding times and parametrization semantics in a range of representative languages and have identified a common ground that can be suitably mapped to different lan ...

Keywords: antiunification, generics, parametric polymorphism, software component architecture, templates

19 A comparative study of language support for generic programming

Ronald Garcia, Jaakko Jarvi, Andrew Lumsdaine, Jeremy G. Siek, Jeremiah Willcock

October 2003 **ACM SIGPLAN Notices , Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications OOPSLA '03**, Volume 38 Issue 11

Publish r: ACM Press

Full text available: pdf(237.38 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Many modern programming languages support basic generic programming, sufficient to implement type-safe polymorphic containers. Some languages have moved beyond this basic support to a broader, more powerful interpretation of generic programming, and

their extensions have proven valuable in practice. This paper reports on a comprehensive comparison of generics in six programming languages: C++, Standard ML, Haskell, Eiffel, Java (with its proposed generics extension), and Generic C. By implementi ...

Keyw rds: C#, C++, Eiffel, Haskell, Java, generic programming, generics, polymorphism, standard ML

20 Scalable extensibility via nested inheritance

Nathaniel Nystrom, Stephen Chong, Andrew C. Myers
October 2004 **ACM SIGPLAN Notices , Proceedings of the 19th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '04**, Volume 39 Issue 10
Publisher: ACM Press
Full text available: pdf(196.74 KB) Additional Information: full citation, abstract, references, index terms

Inheritance is a useful mechanism for factoring and reusing code. However, it has limitations for building extensible systems. We describe nested inheritance, a mechanism that addresses some of the limitations of ordinary inheritance and other code reuse mechanisms. Using our experience with an extensible compiler framework, we show how nested inheritance can be used to construct highly extensible software frameworks. The essential aspects of nested inheritance are formalized i ...

Keywords: inheritance, nested classes, object-oriented programming languages, virtual classes

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next